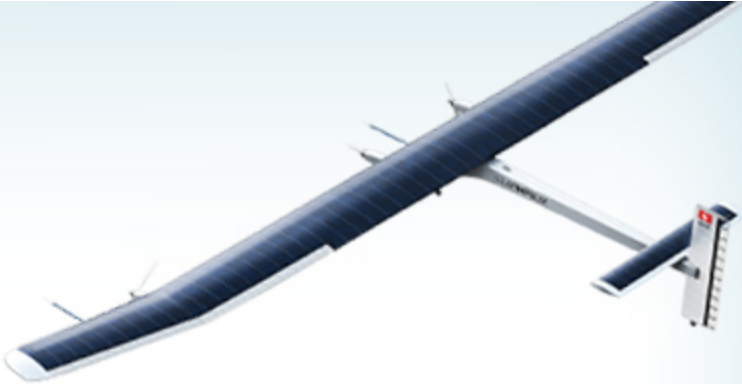


ALTRAN

Global leader in innovation and high-tech engineering consultancy

INNOVATION MAKERS

Formal Verification: Will The Seedling Ever Flower?

Agenda

- Introductions
- Technology Graduation
- Life in Industry
- Asking some Questions…

Agenda

- Introductions
- Technology Graduation
- Life in Industry
- Asking some Questions…

Altran Group

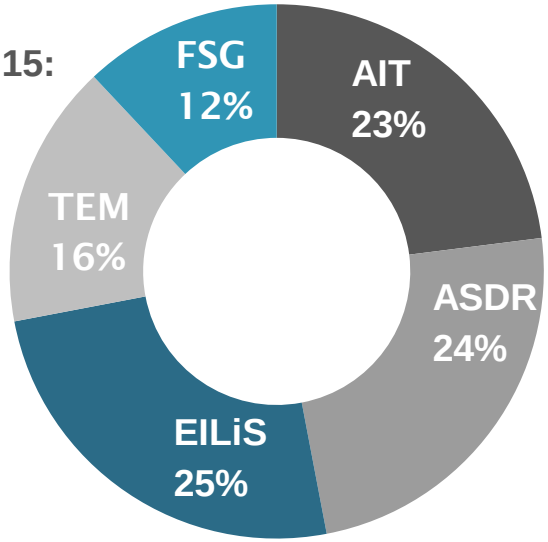
Key numbers (2015)

25,000
+FTEs

€1.945BN
Revenues

Offices in
23 countries

Distribution of Group Revenues 2015:



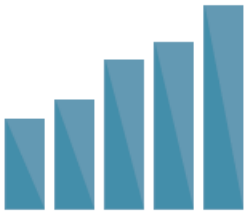
AIT
Automotive, Infrastructure & Transportation
ASDR
Aerospace, Defence & Rail
EILiS
Energy, Industry & Life Sciences
TEM
Telecoms, Electronics & Media
FSG
Financial Services & Government

Altran in the UK

Key numbers (2015)

850+
FTEs*

€159M
Revenues
(2015)*



*Excluding Tessella

Offices in
16 locations

Cambridge Consultants

Tessella

Our World – Critical Software

No defects please!

ALTRAN

Agenda

- Introductions
- Technology Graduation
- Life in Industry
- Asking some Questions…

Avoid introducing defects

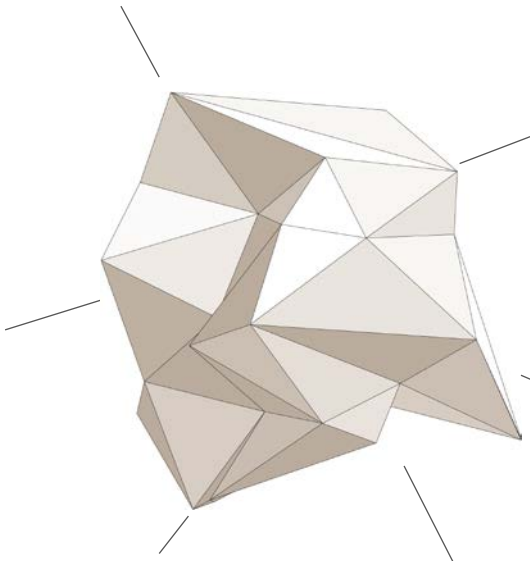
Introducing defects is easy – removing them is hard, and expensive.

Remove defects early

Defects removed early when changes are cheap.

Generate evidence as you go

Evidence needed for certification is produced naturally as a by-product of the process.



Testing is a demonstration of correctness

Not the point where we start debugging.

Prediction over observation.

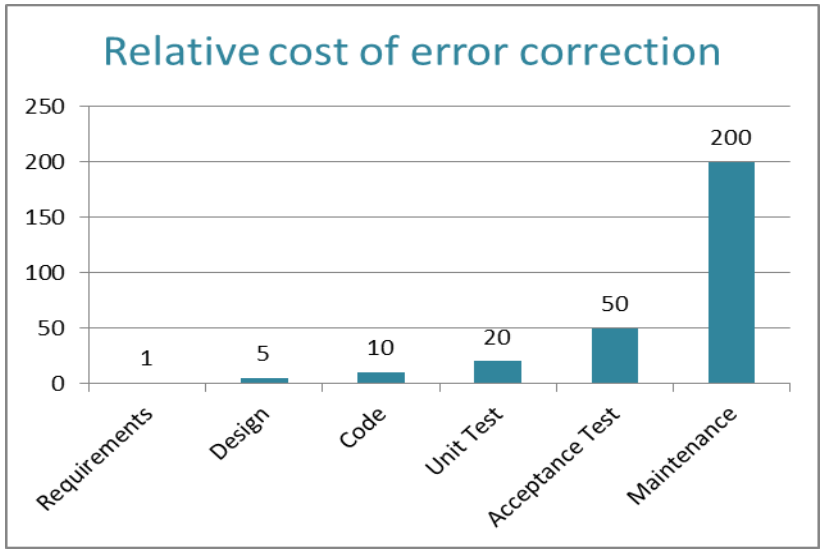
Better can be cheaper

Safety is given. How you get there determines the cost.

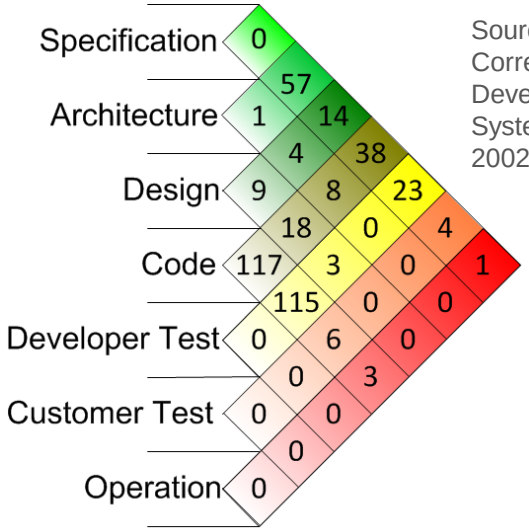
Zero tolerance of defects

We cannot claim zero defects but we can have a zero tolerance attitude to them.

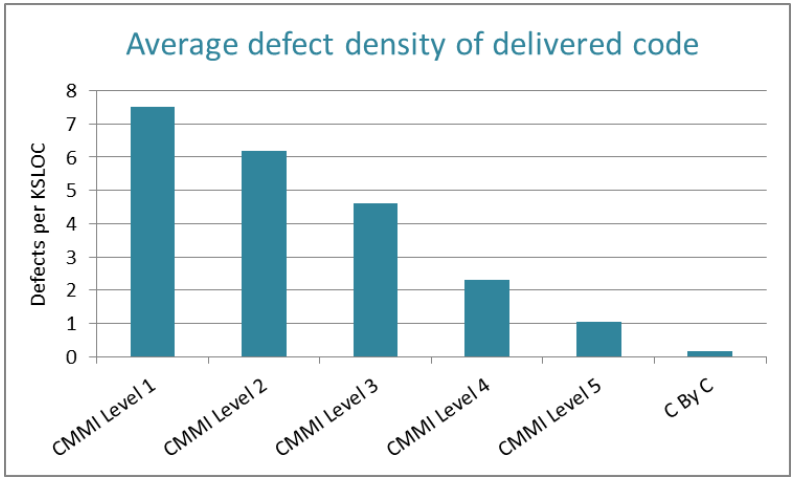
The cost of errors



Source:Leffingwell
http://www.rational.com/media/whitepapers/roi1.pdf



Source: IEEE Software. Correctness by Construction: Developing a Commercial Secure System, Hall and Chapman, Jan 2002



Source: CMM Data from Jones, Caspers: Software Assessments, Benchmarks and Best Practices. Reading, MA: Addison-Wesley, 2002

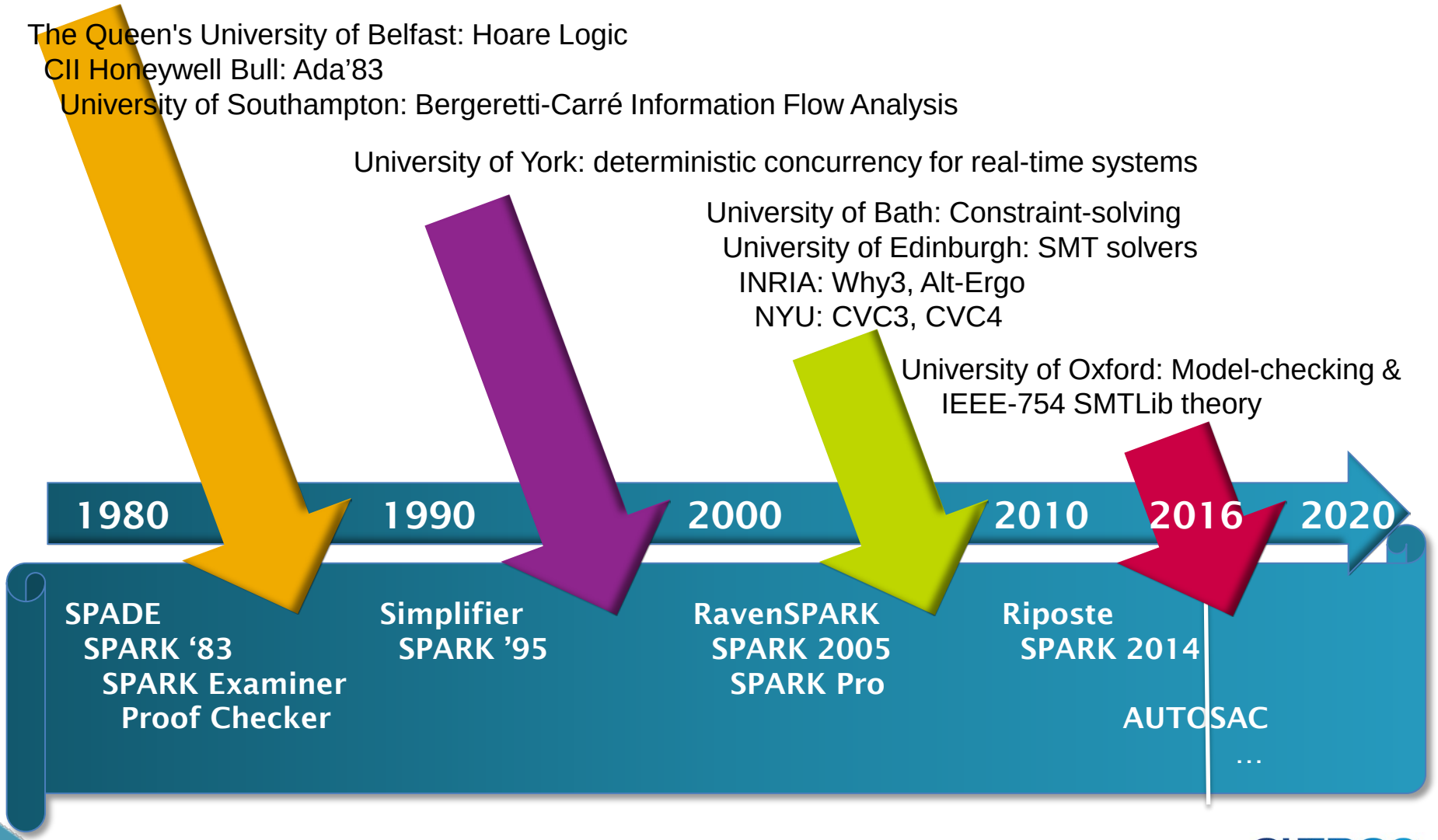
Source: C By C data from Correctness by Construction: A manifesto for High-Integrity software, Croxford and Chapman 2005

ALTRAN

What's in the toolbox?

- Lots of things!
  - REVEAL® (Jackson style D, S, R) requirements
  - Z
  - SCADE
  - Matlab / QGen
  - SPARK
  - ConTestor
  - High Integrity Agile
  - etc.

SPARK: Technology Transfer Timeline



ALTRAN

Agenda

- Introductions
- Technology Graduation
- Life in Industry
- Asking some Questions…

SHOLIS

Ultra ELECTRONICS

CONTEXT & OBJECTIVES

- A military system that displays whether ship and helicopter parameters are within safe landing limits.
- UK MOD required the system certified to Def Stan 00-55.
- Def Stan 00-55 required full functional proof.
- First software ever developed to this standard.

APPROACH & SOLUTION

- Specification written in Z.
- Z type checking performed.
- Code developed in SPARK.
- Z specification translated to SPARK specifications.
- Code proven to be compliant with SPARK specifications.

RESULTS & ADDED VALUE

- System passed as Def Stan 00-55 compliant.
- 42 kloc / 9000 VCs.
- 0.22 defects per kloc.
- Demonstrated low value of unit testing when formal methods used.

ALTRAN

MGKC

CONTEXT & OBJECTIVES

- Smart card security
- Flaws in software could lead to very high financial impact and reduced confidence in the product
- Security standard ITSEC E6

APPROACH & SOLUTION

- Specification written in Z
- Z type checking performed
- Code developed in SPARK
- Security properties translated to SPARK specifications
- Code proven to maintain security properties

RESULTS & ADDED VALUE

- 100,000 lines of SPARK, Ada, C, C++ and SQL
- Three trivial defects, one spec defect – fixed under warranty in first year of operation
- 0.04 defects per kloc

EMU

CONTEXT & OBJECTIVES

- To provide a first class service the supplier needs a state of the art Engine Health Monitoring system
- Engine monitoring units needed for whole engine family
- Each engine type has different hardware considerations and electronic interfaces

APPROACH & SOLUTION

- C By C deployed.
- Supported systems engineering and requirements development
- Software design using Informed methodology
- Software developed using SPARK technologies
- Proof of absence of run time errors.

RESULTS & ADDED VALUE

- Compliant with DO-178B Level C
- Family of engines supported: common source code that is verified once used often
- Joint research project to develop next generation EHM (adaptive, 2-way comms)

ALTRAN

Tokeneer Demonstrator

"Produces code more quickly and reliably and at lower cost than traditional methods", **NSA**

CONTEXT & OBJECTIVES

- US National Security Agency leads the US government in cryptology
- To understand how to build systems that are:
  - cost-effective
  - ultra secure
  - certifiable to Common Criteria EAL5.
- Tokeneer is a biometric access control system

APPROACH & SOLUTION

- Specification written in Z
- Security properties captured in SPARK contracts
- Code written in SPARK
- Security properties proven

RESULTS & ADDED VALUE

- Compliant with Common Criteria EAL5
- Zero defects found in independent system test
- 10kloc SPARK, producing 2623 VCs
- 2513 proved automatically (95.8%)
- Open source information at http://www.adacore.com/tokeneer

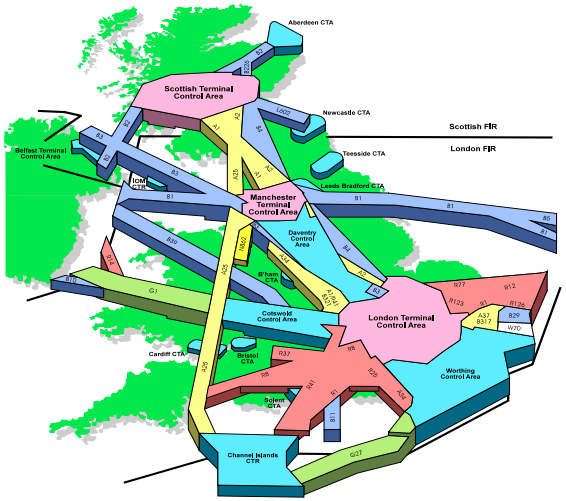
ALTRAN

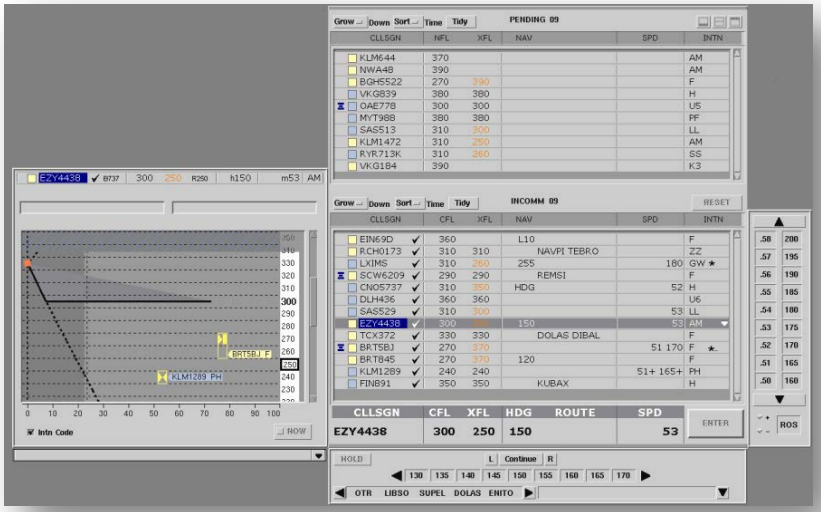
iFACTS

NATS

1. iFACTS enables controllers to handle more traffic safely
2. It increases 'look ahead' from 2 to 15 minutes
3. Provides controller tools
   - » Medium Term Conflict Detection (MTCD)
   - » Trajectory Prediction (TP)
   - » Monitoring aids
4. Altran appointed to develop new software for iFACTS

   Needs to meet CAA's stringent SW01 objectives





- » In full operational service since December 2011.
- » Formal functional specification in Z.
- » Almost all code in SPARK – 250kloc logical.
- » Proved "type (and memory) safe" – i.e. for any input data and state, no undefined behaviour, no crashes, no exceptions.
- » 152,927 VCs, of which 98.76% discharged automatically. User-defined lemmas and review for the remainder.

""The iFACTS system and operational concept is ground breaking and genuinely unique in the world of Air Traffic Control. The new working process is already seeing significant benefits across the NATS business, and airports and airline customers are seeing the benefit too."

Jonathan Astill, General Manager, Area Control, NATS

altran

Agenda

- Introductions
- Technology Graduation
- Life in Industry
- Asking some Questions…

Discussion

- So why haven't formal methods taken over the software industry?
  - They lead to cheaper projects
  - They lead to higher quality projects
- What more do you need?
- Why do we focus on critical software?

Objections

"I don't want to be locked into a tool from a single vendor."

- Actually I have some sympathy with this one.
- But it's not an issue limited to formal methods.
- And it's not such a big deal as you might think because all projects freeze tools early on.

Objections

"I've bought <tool> and it was very expensive so I have to use it."

- This is what happens when finance run projects instead of engineers.
- Inhibits innovation, research, improvement, and onward development.

Objections

"My team don't know <tool> so we can't use it."

- If your team have a good grounding in basic computer science principles, then given the right training, they can pick up any tool quick enough.
- Another inhibitor to innovation.

Objections

"We want to use Industry Standards" or
"We want to use Industry Practice."

- You should use Best Practice.

Objections

"We don't like to spend more upfront."

- Generally the cost profile of a formal methods project has more spend before code starts to be written.
- But all the data shows the spend overall is lower.

Objections

"I want a sexy drag-and-drop graphical interface."

- You are shallow and vacuous.
- Tools exist; usability will follow users.

Discussion

I conclude that industry rejection of formal methods is not a logical position … but how do you combat that?

- Tougher standards?
- End user education?
- Hide the formality?

ConTestor

- We are currently rolling out a new test approach as part of our verification toolset.
- Many teams automate the running of tests.
- We are automating the initial production of tests too.
- It's a hidden formal method.
- Is hiding the formality the way forward?

  I don't know … but nor do I currently know anything better…

So what have we learnt?

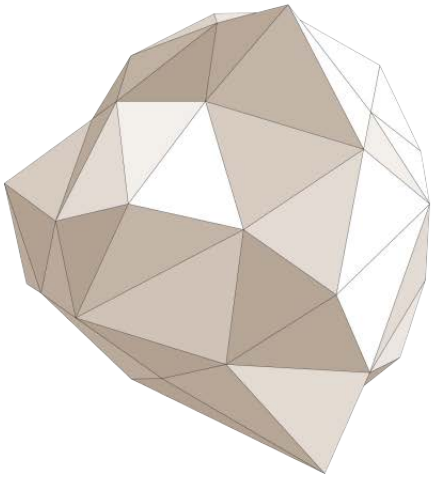
- University research in formal methods can be deployed with great success in industrial projects.
- Getting industry acceptance is the hardest part.
- It's not a logical rejection.
- It's not clear what the objection is.

What must we do?

- Stay Logical: We always need independent up-to-date papers comparing formal and non-formal approaches so that we can have logical, data-based, discussions.

- Fight the illogical: We need to bring formal methods to the attention of industry in new ways.

INNOVATION MAKERS



altran